Hashing to Elliptic Curves and Cryptanalysis of RSA-Based Schemes

Mehdi Tibouchi

École normale supérieure

Ph.D. Defense
2011–09–23

Cryptology

Cryptology is the science of secret messages.
It has two opposite, complementary sides.

Cryptology

Cryptology is the science of secret messages.
It has two opposite, complementary sides.

Cryptography

Constructing systems to ensure various security properties of communications.

[redacted]

Cryptology

Cryptology is the science of secret messages.
It has two opposite, complementary sides.

Cryptography

Constructing systems to ensure various security properties of communications.

[redacted]

Cryptology

Cryptology is the science of secret messages.
It has two opposite, complementary sides.

Cryptography

Constructing systems to ensure various security properties of communications.

Cryptology

Cryptology is the science of secret messages.
It has two opposite, complementary sides.

Cryptography

Constructing systems to ensure various security properties of communications.

Cryptanalysis

Uncovering flaws in those systems so as to break the security of communications.

Ensuring confidentiality

Give me that pencil, will you?

Ensuring confidentiality

Give me that pencil, will you? →

Ensuring confidentiality

Give me that pencil, will you?

!!!

Ensuring confidentiality

5809207b5d4cf644b9fecee81ab7fb8 →

???

Ensuring confidentiality

5809207b5d4cf644b9fecee81ab7fb8

???

Ensuring confidentiality

5809207b5d4cf644b9fecee81ab7fb8

???

Security properties

Besides confidentiality (which is traditional goal of cryptography), other security properties can be sought, such as:

- authenticity: whether it really is Alice talking;
- integrity: whether the message is what was actually sent;
- non-repudiation: so Alice cannot claim she didn't write the message;
- and many more...

Security properties

Besides confidentiality (which is traditional goal of cryptography), other security properties can be sought, such as:

- authenticity: whether it really is Alice talking;
- integrity: whether the message is what was actually sent;
- non-repudiation: so Alice cannot claim she didn't write the message;
- and many more...

Security properties

Besides confidentiality (which is traditional goal of cryptography), other security properties can be sought, such as:

- authenticity: whether it really is Alice talking;
- integrity: whether the message is what was actually sent;
- non-repudiation: so Alice cannot claim she didn't write the message;
- and many more...

Security properties

Besides confidentiality (which is traditional goal of cryptography), other security properties can be sought, such as:

- authenticity: whether it really is Alice talking;
- integrity: whether the message is what was actually sent;
- non-repudiation: so Alice cannot claim she didn't write the message;
- and many more...

Cryptography vs. cryptanalysis

Cryptography

Cryptanalysis

Cryptography vs. cryptanalysis

Cryptography

Construct secure schemes

Cryptanalysis

Cryptography vs. cryptanalysis

Cryptography

Construct secure schemes

Cryptanalysis

Break those schemes

Cryptography vs. cryptanalysis

Cryptography

Construct secure schemes
Prove security properties in a certain model

Cryptanalysis

Break those schemes

Cryptography vs. cryptanalysis

Cryptography

Construct secure schemes

Prove security properties in a certain model

Cryptanalysis

Break those schemes

Circumvent the model

Cryptography vs. cryptanalysis

Cryptography

Construct secure schemes

Prove security properties in a certain model

...based on given hardness assumptions

Cryptanalysis

Break those schemes

Circumvent the model

Cryptography vs. cryptanalysis

Cryptography

Construct secure schemes

Prove security properties in a certain model

...based on given hardness assumptions

Cryptanalysis

Break those schemes

Circumvent the model

Show that the “hard” problems are not that hard

Cryptography vs. cryptanalysis

Cryptography

Construct secure schemes

Prove security properties in a certain model

...based on given hardness assumptions

Implement the schemes in applications

Cryptanalysis

Break those schemes

Circumvent the model

Show that the “hard” problems are not that hard

Cryptography vs. cryptanalysis

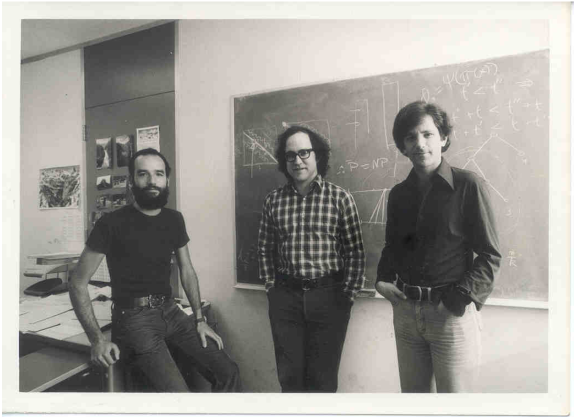
Cryptography	Cryptanalysis
Construct secure schemes	Break those schemes
Prove security properties in a certain model	Circumvent the model
...based on given hardness assumptions	Show that the "hard" problems are not that hard
Implement the schemes in applications	Tamper with the implementation

Outline

Outline

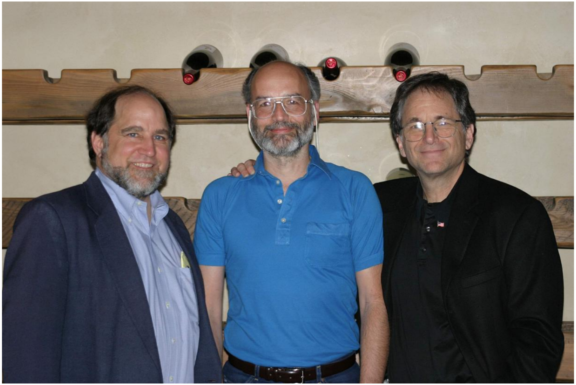
RSA

In 1976, Rivest, Shamir and Adleman proposed the first construction of a public-key encryption scheme and of a digital signature scheme.



RSA

In 1976, Rivest, Shamir and Adleman proposed the first construction of a public-key encryption scheme and of a digital signature scheme.



RSA signatures (textbook ver.)

- "Public key": to authenticate herself to Bob, Alice doesn't need to share a secret with him. She can sign messages and those signatures can be checked by anyone.
- The scheme is as follows:
  - Key generation: Alice picks random large primes p, q and computes $N = pq$. She chooses e coprime to $\varphi(N) = (p-1)(q-1)$, and computes d the inverse of $e \bmod \varphi(N)$.
    She makes (N, e) public and keeps p, q, d secret.
  - Signature: the signature on a message m is $\sigma = m^d \bmod N$.
  - Verification: to check that the signature σ on m is valid, Bob verifies that $\sigma^e \equiv m \pmod{N}$.
- The scheme is correct, because by Euler's theorem $\sigma^e \equiv m^{ed} \equiv m \pmod{N}$.
- Recovering the secret key d from the public key (N, e) is as hard as factoring the RSA modulus N.

RSA signatures (textbook ver.)

- "Public key": to authenticate herself to Bob, Alice doesn't need to share a secret with him. She can sign messages and those signatures can be checked by anyone.
- The scheme is as follows:
  - Key generation: Alice picks random large primes p, q and computes $N = pq$. She chooses e coprime to $\varphi(N) = (p-1)(q-1)$, and computes d the inverse of $e \bmod \varphi(N)$.
    She makes (N, e) public and keeps p, q, d secret.
  - Signature: the signature on a message m is $\sigma = m^d \bmod N$.
  - Verification: to check that the signature σ on m is valid, Bob verifies that $\sigma^e \equiv m \pmod N$.
- The scheme is correct, because by Euler's theorem $\sigma^e \equiv m^{ed} \equiv m \pmod N$.
- Recovering the secret key d from the public key (N, e) is as hard as factoring the RSA modulus N.

RSA signatures (textbook ver.)

- "Public key": to authenticate herself to Bob, Alice doesn't need to share a secret with him. She can sign messages and those signatures can be checked by anyone.
- The scheme is as follows:
  - Key generation: Alice picks random large primes p, q and computes $N = pq$. She chooses e coprime to $\varphi(N) = (p-1)(q-1)$, and computes d the inverse of $e \bmod \varphi(N)$.
    She makes (N, e) public and keeps p, q, d secret.
  - Signature: the signature on a message m is $\sigma = m^d \bmod N$.
  - Verification: to check that the signature σ on m is valid, Bob verifies that $\sigma^e \equiv m \pmod N$.
- The scheme is correct, because by Euler's theorem $\sigma^e \equiv m^{ed} \equiv m \pmod N$.
- Recovering the secret key d from the public key (N, e) is as hard as factoring the RSA modulus N.

RSA signatures (textbook ver.)

- "Public key": to authenticate herself to Bob, Alice doesn't need to share a secret with him. She can sign messages and those signatures can be checked by anyone.
- The scheme is as follows:
  - Key generation: Alice picks random large primes p, q and computes $N = pq$. She chooses e coprime to $\varphi(N) = (p-1)(q-1)$, and computes d the inverse of $e \bmod \varphi(N)$.
    She makes (N, e) public and keeps p, q, d secret.
  - Signature: the signature on a message m is $\sigma = m^d \bmod N$.
  - Verification: to check that the signature σ on m is valid, Bob verifies that $\sigma^e \equiv m \pmod{N}$.
- The scheme is correct, because by Euler's theorem $\sigma^e \equiv m^{ed} \equiv m \pmod{N}$.
- Recovering the secret key d from the public key (N, e) is as hard as factoring the RSA modulus N.

The importance of padding functions

- As such, the scheme is not secure. For example, if Alice publishes signatures σ_1, σ_2 on messages m_1, m_2, then anyone can forge a signature on the product $m_1 \cdot m_2$: simply $\sigma = \sigma_1 \cdot \sigma_2 \bmod N$.
- The usual solution is to apply the RSA function not to m itself but to $\mu(m)$ for some public function μ, called a padding:

$$\sigma = \mu(m)^d \bmod N$$

- In applications until the 1990s, μ was constructed to be fast and thwart some known attacks, but with no proof of security: ad-hoc paddings, many of which have been shown to be flawed (example in this thesis).
- Recently, provably secure paddings have been constructed, at least in the idealized "random oracle model". For example, the RSA signature scheme obtained by choosing μ as a full-length random oracle (FDH) is secure.

The importance of padding functions

- As such, the scheme is not secure. For example, if Alice publishes signatures σ_1, σ_2 on messages m_1, m_2, then anyone can forge a signature on the product $m_1 \cdot m_2$: simply $\sigma = \sigma_1 \cdot \sigma_2 \bmod N$.
- The usual solution is to apply the RSA function not to m itself but to $\mu(m)$ for some public function μ, called a padding:

$$\sigma = \mu(m)^d \bmod N$$

- In applications until the 1990s, μ was constructed to be fast and thwart some known attacks, but with no proof of security: ad-hoc paddings, many of which have been shown to be flawed (example in this thesis).
- Recently, provably secure paddings have been constructed, at least in the idealized "random oracle model". For example, the RSA signature scheme obtained by choosing μ as a full-length random oracle (FDH) is secure.

The importance of padding functions

- As such, the scheme is not secure. For example, if Alice publishes signatures σ_1, σ_2 on messages m_1, m_2, then anyone can forge a signature on the product $m_1 \cdot m_2$: simply $\sigma = \sigma_1 \cdot \sigma_2 \bmod N$.
- The usual solution is to apply the RSA function not to m itself but to $\mu(m)$ for some public function μ, called a padding:

$$\sigma = \mu(m)^d \bmod N$$

- In applications until the 1990s, μ was constructed to be fast and thwart some known attacks, but with no proof of security: ad-hoc paddings, many of which have been shown to be flawed (example in this thesis).
- Recently, provably secure paddings have been constructed, at least in the idealized “random oracle model”. For example, the RSA signature scheme obtained by choosing μ as a full-length random oracle (FDH) is secure.

The importance of padding functions

- As such, the scheme is not secure. For example, if Alice publishes signatures σ_1, σ_2 on messages m_1, m_2, then anyone can forge a signature on the product $m_1 \cdot m_2$: simply $\sigma = \sigma_1 \cdot \sigma_2 \bmod N$.
- The usual solution is to apply the RSA function not to m itself but to $\mu(m)$ for some public function μ, called a padding:

$$\sigma = \mu(m)^d \bmod N$$

- In applications until the 1990s, μ was constructed to be fast and thwart some known attacks, but with no proof of security: ad-hoc paddings, many of which have been shown to be flawed (example in this thesis).
- Recently, provably secure paddings have been constructed, at least in the idealized "random oracle model". For example, the RSA signature scheme obtained by choosing μ as a full-length random oracle (FDH) is secure.

Fault attacks

- Security in traditional cryptography: against adversaries that "follow the rules" and try to break a mathematical problem.
- Real-world adversaries want to break a physical cryptographic device.
- Thus, they have more powerful attacks at their disposal.

  Side channels: passively exploit the physical leakage (time, heat, power consumption, etc.) of the device to gain additional information;

  Faults: actively induce device malfunction (power spikes, overheating, laser beams, etc.) to cause exploitable errors in computations.
- Even provably secure schemes like FDH do not necessarily remain secure against such attacks!

Fault attacks

- Security in traditional cryptography: against adversaries that "follow the rules" and try to break a mathematical problem.
- Real-world adversaries want to break a physical cryptographic device.
- Thus, they have more powerful attacks at their disposal.

  Side channels: passively exploit the physical leakage (time, heat, power consumption, etc.) of the device to gain additional information;

  Faults: actively induce device malfunction (power spikes, overheating, laser beams, etc.) to cause exploitable errors in computations.
- Even provably secure schemes like FDH do not necessarily remain secure against such attacks!

Fault attacks

- Security in traditional cryptography: against adversaries that "follow the rules" and try to break a mathematical problem.
- Real-world adversaries want to break a physical cryptographic device.
- Thus, they have more powerful attacks at their disposal.

  Side channels: passively exploit the physical leakage (time, heat, power consumption, etc.) of the device to gain additional information;

  Faults: actively induce device malfunction (power spikes, overheating, laser beams, etc.) to cause exploitable errors in computations.
- Even provably secure schemes like FDH do not necessarily remain secure against such attacks!

Fault attacks

- Security in traditional cryptography: against adversaries that "follow the rules" and try to break a mathematical problem.
- Real-world adversaries want to break a physical cryptographic device.
- Thus, they have more powerful attacks at their disposal.

  Side channels: passively exploit the physical leakage (time, heat, power consumption, etc.) of the device to gain additional information;

  Faults: actively induce device malfunction (power spikes, overheating, laser beams, etc.) to cause exploitable errors in computations.
- Even provably secure schemes like FDH do not necessarily remain secure against such attacks!

Fault attacks

- Security in traditional cryptography: against adversaries that "follow the rules" and try to break a mathematical problem.
- Real-world adversaries want to break a physical cryptographic device.
- Thus, they have more powerful attacks at their disposal.

  Side channels: passively exploit the physical leakage (time, heat, power consumption, etc.) of the device to gain additional information;

  Faults: actively induce device malfunction (power spikes, overheating, laser beams, etc.) to cause exploitable errors in computations.
- Even provably secure schemes like FDH do not necessarily remain secure against such attacks!

Fault attacks

- Security in traditional cryptography: against adversaries that "follow the rules" and try to break a mathematical problem.
- Real-world adversaries want to break a physical cryptographic device.
- Thus, they have more powerful attacks at their disposal.

  Side channels: passively exploit the physical leakage (time, heat, power consumption, etc.) of the device to gain additional information;

  Faults: actively induce device malfunction (power spikes, overheating, laser beams, etc.) to cause exploitable errors in computations.
- Even provably secure schemes like FDH do not necessarily remain secure against such attacks!

Signing with RSA-CRT

- RSA remains the most widely used signature scheme today. It is implemented in many embedded applications (esp. smart cards).
- However, modular exponentiation is rather slow.
- Very commonly used improvement: using the Chinese Remainder Theorem.
  1. $\sigma_p = \mu(m)^d \bmod p$
  2. $\sigma_q = \mu(m)^d \bmod q$
  3. $\sigma = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N$
- Roughly 4-fold speed-up.

Signing with RSA-CRT

- RSA remains the most widely used signature scheme today. It is implemented in many embedded applications (esp. smart cards).
- However, modular exponentiation is rather slow.
- Very commonly used improvement: using the Chinese Remainder Theorem.
  1. $\sigma_p = \mu(m)^d \bmod p$
  2. $\sigma_q = \mu(m)^d \bmod q$
  3. $\sigma = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N$
- Roughly 4-fold speed-up.

Signing with RSA-CRT

- RSA remains the most widely used signature scheme today. It is implemented in many embedded applications (esp. smart cards).
- However, modular exponentiation is rather slow.
- Very commonly used improvement: using the Chinese Remainder Theorem.
  1. $\sigma_p = \mu(m)^d \bmod p$
  2. $\sigma_q = \mu(m)^d \bmod q$
  3. $\sigma = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N$
- Roughly 4-fold speed-up.

Signing with RSA-CRT

- RSA remains the most widely used signature scheme today. It is implemented in many embedded applications (esp. smart cards).
- However, modular exponentiation is rather slow.
- Very commonly used improvement: using the Chinese Remainder Theorem.
  1. $\sigma_p = \mu(m)^d \bmod p$
  2. $\sigma_q = \mu(m)^d \bmod q$
  3. $\sigma = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N$
- Roughly 4-fold speed-up.

The Boneh-DeMillo-Lipton fault attack (1997)

- The problem with CRT: fault attacks.
- A fault in signature generation makes it possible to recover the secret key!
  1. $\sigma_p = \mu(m)^d \bmod p$
  2. $\sigma'_q \neq \mu(m)^d \bmod q$ ← fault
  3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma'_q) \bmod N$ ← faulty signature
- Then ${\sigma'}^e$ is $\mu(m) \bmod p$ but not mod q, so the attacker can then factor N:

$$p = \gcd({\sigma'}^e - \mu(m), N)$$

- This attack applies to any deterministic padding, including provably secure ones like FDH.

The Boneh-DeMillo-Lipton fault attack (1997)

- The problem with CRT: fault attacks.
- A fault in signature generation makes it possible to recover the secret key!
  1. $\sigma_p = \mu(m)^d \bmod p$
  2. $\sigma'_q \neq \mu(m)^d \bmod q$ ← fault
  3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma'_q) \bmod N$ ← faulty signature
- Then ${\sigma'}^e$ is $\mu(m) \bmod p$ but not mod q, so the attacker can then factor N:

$$p = \gcd({\sigma'}^e - \mu(m), N)$$

- This attack applies to any deterministic padding, including provably secure ones like FDH.

The Boneh-DeMillo-Lipton fault attack (1997)

- The problem with CRT: fault attacks.
- A fault in signature generation makes it possible to recover the secret key!
    1. $\sigma_p = \mu(m)^d \bmod p$
    2. $\sigma'_q \neq \mu(m)^d \bmod q$ ← fault
    3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma'_q) \bmod N$ ← faulty signature
- Then σ'^e is $\mu(m) \bmod p$ but not mod q, so the attacker can then factor N:

$$p = \gcd(\sigma'^e - \mu(m), N)$$

- This attack applies to any deterministic padding, including provably secure ones like FDH.

The Boneh-DeMillo-Lipton fault attack (1997)

- The problem with CRT: fault attacks.
- A fault in signature generation makes it possible to recover the secret key!
  1. $\sigma_p = \mu(m)^d \bmod p$
  2. $\sigma'_q \neq \mu(m)^d \bmod q$ ← fault
  3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma'_q) \bmod N$ ← faulty signature
- Then ${\sigma'}^e$ is $\mu(m) \bmod p$ but not mod q, so the attacker can then factor N:

$$p = \gcd({\sigma'}^e - \mu(m), N)$$

- This attack applies to any deterministic padding, including provably secure ones like FDH.

The Boneh-DeMillo-Lipton fault attack (1997)

- The problem with CRT: fault attacks.
- A fault in signature generation makes it possible to recover the secret key!
  1. $\sigma_p = \mu(m)^d \bmod p$
  2. $\sigma'_q \neq \mu(m)^d \bmod q$ ← fault
  3. $\sigma' = \text{CRT}(\sigma_p, \sigma'_q) \bmod N$ ← faulty signature
- Then σ'^e is $\mu(m) \bmod p$ but not mod q, so the attacker can then factor N:

$$p = \gcd(\sigma'^e - \mu(m), N)$$

- This attack applies to any deterministic padding, including provably secure ones like FDH.

The Boneh-DeMillo-Lipton fault attack (1997)

- The problem with CRT: fault attacks.
- A fault in signature generation makes it possible to recover the secret key!
  1. $\sigma_p = \mu(m)^d \bmod p$
  2. $\sigma'_q \neq \mu(m)^d \bmod q$ ← fault
  3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma'_q) \bmod N$ ← faulty signature
- Then σ'^e is $\mu(m)$ mod p but not mod q, so the attacker can then factor N:

$$p = \gcd(\sigma'^e - \mu(m), N)$$

- This attack applies to any deterministic padding, including provably secure ones like FDH.

The Boneh-DeMillo-Lipton fault attack (1997)

- The problem with CRT: fault attacks.
- A fault in signature generation makes it possible to recover the secret key!
  1. $\sigma_p = \mu(m)^d \bmod p$
  2. $\sigma'_q \neq \mu(m)^d \bmod q$ ← fault
  3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma'_q) \bmod N$ ← faulty signature
- Then ${\sigma'}^e$ is $\mu(m)$ mod p but not mod q, so the attacker can then factor N:

$$p = \gcd({\sigma'}^e - \mu(m), N)$$

- This attack applies to any deterministic padding, including provably secure ones like FDH.

Shamir's trick

- Faults against RSA-CRT signatures have been an active research subject since then. Many variants and countermeasures have been proposed.
- One simple countermeasure due to Shamir is to compute the signature as follows (r is a small fixed integer like $2^{31}-1$):
  1. $\sigma_p^+ = \mu(m)^d \bmod r \cdot p$
  2. $\sigma_q^+ = \mu(m)^d \bmod r \cdot q$
  3. if $\sigma_p^+ \not\equiv \sigma_q^+ \pmod r$, abort
  4. $\sigma = \mathrm{CRT}(\sigma_p^+, \sigma_q^+) \bmod N$
- If one of the half-exponentiations is perturbed, signature generation is very likely to abort, and hence the fault attacker cannot factor anymore!

Shamir's trick

- Faults against RSA-CRT signatures have been an active research subject since then. Many variants and countermeasures have been proposed.
- One simple countermeasure due to Shamir is to compute the signature as follows (r is a small fixed integer like $2^{31} - 1$):
  1. $\sigma_p^+ = \mu(m)^d \bmod r \cdot p$
  2. $\sigma_q^+ = \mu(m)^d \bmod r \cdot q$
  3. if $\sigma_p^+ \not\equiv \sigma_q^+ \pmod r$, abort
  4. $\sigma = \mathrm{CRT}(\sigma_p^+, \sigma_q^+) \bmod N$
- If one of the half-exponentiations is perturbed, signature generation is very likely to abort, and hence the fault attacker cannot factor anymore!

Shamir's trick

- Faults against RSA-CRT signatures have been an active research subject since then. Many variants and countermeasures have been proposed.
- One simple countermeasure due to Shamir is to compute the signature as follows (r is a small fixed integer like $2^{31} - 1$):
  1. $\sigma_p^+ = \mu(m)^d \bmod r \cdot p$
  2. $\sigma_q^+ = \mu(m)^d \bmod r \cdot q$
  3. if $\sigma_p^+ \not\equiv \sigma_q^+ \pmod r$, abort
  4. $\sigma = \mathrm{CRT}(\sigma_p^+, \sigma_q^+) \bmod N$
- If one of the half-exponentiations is perturbed, signature generation is very likely to abort, and hence the fault attacker cannot factor anymore!

Outline

Introduction

RSA Cryptanalysis

- RSA-CRT signatures
- Modulus fault attacks

Hashing to Elliptic Curves

- Elliptic curve cryptography
- Hashing to elliptic curves
- Constructing good hash functions

Attacking the modulus

- A lot of work has been invested into protecting the exponentiations in RSA-CRT signature generation.
- So what about attacking another part of the algorithm?
- Idea: attack the modular reduction instead!
  1. $\sigma_p = \mu(m)^d \bmod p \quad \leftarrow$ correct
  2. $\sigma_q = \mu(m)^d \bmod q \quad \leftarrow$ correct
  3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N' \quad \leftarrow$ faulty signature: wrong modular reduction!
- This new, strange type of faults can also be used to factor N.

Attacking the modulus

- A lot of work has been invested into protecting the exponentiations in RSA-CRT signature generation.
- So what about attacking another part of the algorithm?
- Idea: attack the modular reduction instead!
  1. $\sigma_p = \mu(m)^d \bmod p$ ← correct
  2. $\sigma_q = \mu(m)^d \bmod q$ ← correct
  3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N'$ ← faulty signature: wrong modular reduction!
- This new, strange type of faults can also be used to factor N.

Attacking the modulus

- A lot of work has been invested into protecting the exponentiations in RSA-CRT signature generation.
- So what about attacking another part of the algorithm?
- Idea: attack the modular reduction instead!
  1. $\sigma_p = \mu(m)^d \bmod p$ ← correct
  2. $\sigma_q = \mu(m)^d \bmod q$ ← correct
  3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N'$ ← faulty signature: wrong modular reduction!
- This new, strange type of faults can also be used to factor N.

Attacking the modulus

- A lot of work has been invested into protecting the exponentiations in RSA-CRT signature generation.
- So what about attacking another part of the algorithm?
- Idea: attack the modular reduction instead!
  1. $\sigma_p = \mu(m)^d \bmod p$ ← correct
  2. $\sigma_q = \mu(m)^d \bmod q$ ← correct
  3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N'$ ← faulty signature: wrong modular reduction!
- This new, strange type of faults can also be used to factor N.

Attacking the modulus

- A lot of work has been invested into protecting the exponentiations in RSA-CRT signature generation.
- So what about attacking another part of the algorithm?
- Idea: attack the modular reduction instead!
  1. $\sigma_p = \mu(m)^d \bmod p$ ← correct
  2. $\sigma_q = \mu(m)^d \bmod q$ ← correct
  3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N'$ ← faulty signature: wrong modular reduction!
- This new, strange type of faults can also be used to factor N.

Attacking the modulus

- A lot of work has been invested into protecting the exponentiations in RSA-CRT signature generation.
- So what about attacking another part of the algorithm?
- Idea: attack the modular reduction instead!
  1. $\sigma_p = \mu(m)^d \bmod p$ ← correct
  2. $\sigma_q = \mu(m)^d \bmod q$ ← correct
  3. $\sigma' = \text{CRT}(\sigma_p, \sigma_q) \bmod N'$ ← faulty signature: wrong modular reduction!
- This new, strange type of faults can also be used to factor N.

Attacking the modulus

- A lot of work has been invested into protecting the exponentiations in RSA-CRT signature generation.
- So what about attacking another part of the algorithm?
- Idea: attack the modular reduction instead!
  1. $\sigma_p = \mu(m)^d \bmod p$ ← correct
  2. $\sigma_q = \mu(m)^d \bmod q$ ← correct
  3. $\sigma' = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N'$ ← faulty signature: wrong modular reduction!
- This new, strange type of faults can also be used to factor N.

Using the fault (I)

- More precisely, suppose we can obtain the same signature on a certain message twice, once correctly and once with a fault. Then we get:

$$\begin{cases} \sigma = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N & \leftarrow \text{correct} \\ \sigma' = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N' & \leftarrow \text{faulty} \end{cases}$$

- Applying the CRT to these two relations, we obtain the value $\mathrm{CRT}(\sigma_p, \sigma_q) \bmod NN'$.
- Now recall that:

$$\mathrm{CRT}(\sigma_p, \sigma_q) = \alpha \cdot \sigma_p + \beta \cdot \sigma_q$$

where

$$\alpha = q \cdot (q^{-1} \bmod p) \quad \beta = p \cdot (p^{-1} \bmod q)$$

- In particular, $\mathrm{CRT}(\sigma_p, \sigma_q)$ is an integer of size $\approx N^{3/2}$, so if we know it modulo $NN' \approx N^2$, we actually know its value in $\mathbb{Z}$.

Using the fault (I)

- More precisely, suppose we can obtain the same signature on a certain message twice, once correctly and once with a fault. Then we get:

$$\begin{cases} \sigma = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N & \leftarrow \text{correct} \\ \sigma' = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N' & \leftarrow \text{faulty} \end{cases}$$

- Applying the CRT to these two relations, we obtain the value $\mathrm{CRT}(\sigma_p, \sigma_q) \bmod NN'$.
- Now recall that:

$$\mathrm{CRT}(\sigma_p, \sigma_q) = \alpha \cdot \sigma_p + \beta \cdot \sigma_q$$

where

$$\alpha = q \cdot (q^{-1} \bmod p) \quad \beta = p \cdot (p^{-1} \bmod q)$$

- In particular, $\mathrm{CRT}(\sigma_p, \sigma_q)$ is an integer of size $\approx N^{3/2}$, so if we know it modulo $NN' \approx N^2$, we actually know its value in $\mathbb{Z}$.

Using the fault (I)

- More precisely, suppose we can obtain the same signature on a certain message twice, once correctly and once with a fault. Then we get:

$$\begin{cases} \sigma = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N & \leftarrow \text{correct} \\ \sigma' = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N' & \leftarrow \text{faulty} \end{cases}$$

- Applying the CRT to these two relations, we obtain the value $\mathrm{CRT}(\sigma_p, \sigma_q) \bmod NN'$.
- Now recall that:

$$\mathrm{CRT}(\sigma_p, \sigma_q) = \alpha \cdot \sigma_p + \beta \cdot \sigma_q$$

where

$$\alpha = q \cdot (q^{-1} \bmod p) \quad \beta = p \cdot (p^{-1} \bmod q)$$

- In particular, $\mathrm{CRT}(\sigma_p, \sigma_q)$ is an integer of size $\approx N^{3/2}$, so if we know it modulo $NN' \approx N^2$, we actually know its value in $\mathbb{Z}$.

Using the fault (I)

- More precisely, suppose we can obtain the same signature on a certain message twice, once correctly and once with a fault. Then we get:

$$\begin{cases} \sigma = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N & \leftarrow \text{correct} \\ \sigma' = \mathrm{CRT}(\sigma_p, \sigma_q) \bmod N' & \leftarrow \text{faulty} \end{cases}$$

- Applying the CRT to these two relations, we obtain the value $\mathrm{CRT}(\sigma_p, \sigma_q) \bmod NN'$.
- Now recall that:

$$\mathrm{CRT}(\sigma_p, \sigma_q) = \alpha \cdot \sigma_p + \beta \cdot \sigma_q$$

where

$$\alpha = q \cdot (q^{-1} \bmod p) \quad \beta = p \cdot (p^{-1} \bmod q)$$

- In particular, $\mathrm{CRT}(\sigma_p, \sigma_q)$ is an integer of size $\approx N^{3/2}$, so if we know it modulo $NN' \approx N^2$, we actually know its value in $\mathbb{Z}$.

Using the fault (II)

Each pair formed of a correct and of a faulty signature gives us an equation of the form:

$$v = \alpha \cdot x + \beta \cdot y$$

where v is known, α, β are unknown, fixed and of size N, and x, y are unknown, of size $N^{1/2}$, and depend on the signature.

One such relation doesn't get us far, but since (x, y) is small compared to (α, β), we expect multiple relations of this form to allow us to recover the x's and y's, and hence factor N.

So suppose we can obtain a vector $\mathbf{v}$ of ℓ CRT values, so that we have an equation:

$$\mathbf{v} = \alpha\mathbf{x} + \beta\mathbf{y}$$

The goal is to recover $\mathbf{x}$ and $\mathbf{y}$ from $\mathbf{v}$. To do so, we can used a cryptanlytic technique introduced by Nguyen and Stern in the 1990s: orthogonal lattices.

Using the fault (II)

Each pair formed of a correct and of a faulty signature gives us an equation of the form:

$$v = \alpha \cdot x + \beta \cdot y$$

where v is known, α, β are unknown, fixed and of size N, and x, y are unknown, of size $N^{1/2}$, and depend on the signature.

One such relation doesn't get us far, but since (x, y) is small compared to (α, β), we expect multiple relations of this form to allow us to recover the x's and y's, and hence factor N.

So suppose we can obtain a vector $\mathbf{v}$ of ℓ CRT values, so that we have an equation:

$$\mathbf{v} = \alpha \mathbf{x} + \beta \mathbf{y}$$

The goal is to recover $\mathbf{x}$ and $\mathbf{y}$ from $\mathbf{v}$. To do so, we can used a cryptanlytic technique introduced by Nguyen and Stern in the 1990s: orthogonal lattices.

Using the fault (II)

Each pair formed of a correct and of a faulty signature gives us an equation of the form:

$$v = \alpha \cdot x + \beta \cdot y$$

where v is known, α, β are unknown, fixed and of size N, and x, y are unknown, of size $N^{1/2}$, and depend on the signature.

One such relation doesn't get us far, but since (x, y) is small compared to (α, β), we expect multiple relations of this form to allow us to recover the x's and y's, and hence factor N.

So suppose we can obtain a vector $\mathbf{v}$ of ℓ CRT values, so that we have an equation:

$$\mathbf{v} = \alpha\mathbf{x} + \beta\mathbf{y}$$

The goal is to recover $\mathbf{x}$ and $\mathbf{y}$ from $\mathbf{v}$. To do so, we can used a cryptanlytic technique introduced by Nguyen and Stern in the 1990s: orthogonal lattices.

Lattice attack overview

- Compute a reduced basis $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-1})$ of the lattice of vectors in $\mathbb{Z}^\ell$ orthogonal to $\mathbf{v}$.
- Since $\mathbf{v} = \alpha\mathbf{x} + \beta\mathbf{y}$, the $\mathbf{b_i}$'s satisfy:

$$\alpha\langle \mathbf{b}_i, \mathbf{x}\rangle + \beta\langle \mathbf{b}_i, \mathbf{y}\rangle = 0$$

- But the smallest nonzero solution (s, t) to $\alpha s + \beta t = 0$ is of size $\approx N$, so a given $\mathbf{b}_i$ is either orthogonal to both $\mathbf{x}$ and $\mathbf{y}$, or it is of norm $> N^{1/2}$.
- Only $\ell - 2$ independent vectors orthogonal to both $\mathbf{x}$ and $\mathbf{y}$, so $\mathbf{b}_{\ell-1}$ must be of length N. Thus the remaining vectors $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-2})$ form a lattice of volume $\approx N^{3/2}/N^{1/2} = N$. Each of them is heuristically of length $\approx N^{1/(\ell-2)}$. As soon as $\ell \geq 5$, they are of length $\ll N^{1/2}$ and thus orthogonal to $\mathbf{x}, \mathbf{y}$.
- Compute a reduced basis $(\mathbf{x}', \mathbf{y}')$ of the lattice of vectors orthogonal to $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-2})$. The vectors $\mathbf{x}, \mathbf{y}$ are in this lattice, and can be recovered by a quick exhaustive search!

Lattice attack overview

- Compute a reduced basis $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-1})$ of the lattice of vectors in $\mathbb{Z}^\ell$ orthogonal to $\mathbf{v}$.
- Since $\mathbf{v} = \alpha\mathbf{x} + \beta\mathbf{y}$, the $\mathbf{b_i}$'s satisfy:

$$\alpha\langle \mathbf{b}_i, \mathbf{x}\rangle + \beta\langle \mathbf{b}_i, \mathbf{y}\rangle = 0$$

- But the smallest nonzero solution (s, t) to $\alpha s + \beta t = 0$ is of size $\approx N$, so a given $\mathbf{b}_i$ is either orthogonal to both $\mathbf{x}$ and $\mathbf{y}$, or it is of norm $> N^{1/2}$.
- Only $\ell - 2$ independent vectors orthogonal to both $\mathbf{x}$ and $\mathbf{y}$, so $\mathbf{b}_{\ell-1}$ must be of length N. Thus the remaining vectors $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-2})$ form a lattice of volume $\approx N^{3/2}/N^{1/2} = N$. Each of them is heuristically of length $\approx N^{1/(\ell-2)}$. As soon as $\ell \geq 5$, they are of length $\ll N^{1/2}$ and thus orthogonal to $\mathbf{x}, \mathbf{y}$.
- Compute a reduced basis $(\mathbf{x}', \mathbf{y}')$ of the lattice of vectors orthogonal to $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-2})$. The vectors $\mathbf{x}, \mathbf{y}$ are in this lattice, and can be recovered by a quick exhaustive search!

Lattice attack overview

- Compute a reduced basis $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-1})$ of the lattice of vectors in $\mathbb{Z}^\ell$ orthogonal to $\mathbf{v}$.
- Since $\mathbf{v} = \alpha\mathbf{x} + \beta\mathbf{y}$, the $\mathbf{b_i}$'s satisfy:

$$\alpha\langle \mathbf{b}_i, \mathbf{x}\rangle + \beta\langle \mathbf{b}_i, \mathbf{y}\rangle = 0$$

- But the smallest nonzero solution (s, t) to $\alpha s + \beta t = 0$ is of size $\approx N$, so a given $\mathbf{b}_i$ is either orthogonal to both $\mathbf{x}$ and $\mathbf{y}$, or it is of norm $> N^{1/2}$.
- Only $\ell - 2$ independent vectors orthogonal to both $\mathbf{x}$ and $\mathbf{y}$, so $\mathbf{b}_{\ell-1}$ must be of length N. Thus the remaining vectors $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-2})$ form a lattice of volume $\approx N^{3/2}/N^{1/2} = N$. Each of them is heuristically of length $\approx N^{1/(\ell-2)}$. As soon as $\ell \geq 5$, they are of length $\ll N^{1/2}$ and thus orthogonal to $\mathbf{x}, \mathbf{y}$.
- Compute a reduced basis $(\mathbf{x}', \mathbf{y}')$ of the lattice of vectors orthogonal to $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-2})$. The vectors $\mathbf{x}, \mathbf{y}$ are in this lattice, and can be recovered by a quick exhaustive search!

Lattice attack overview

- Compute a reduced basis $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-1})$ of the lattice of vectors in $\mathbb{Z}^\ell$ orthogonal to $\mathbf{v}$.
- Since $\mathbf{v} = \alpha\mathbf{x} + \beta\mathbf{y}$, the $\mathbf{b_i}$'s satisfy:

$$\alpha\langle \mathbf{b}_i, \mathbf{x}\rangle + \beta\langle \mathbf{b}_i, \mathbf{y}\rangle = 0$$

- But the smallest nonzero solution (s, t) to $\alpha s + \beta t = 0$ is of size $\approx N$, so a given $\mathbf{b}_i$ is either orthogonal to both $\mathbf{x}$ and $\mathbf{y}$, or it is of norm $> N^{1/2}$.
- Only $\ell - 2$ independent vectors orthogonal to both $\mathbf{x}$ and $\mathbf{y}$, so $\mathbf{b}_{\ell-1}$ must be of length N. Thus the remaining vectors $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-2})$ form a lattice of volume $\approx N^{3/2}/N^{1/2} = N$. Each of them is heuristically of length $\approx N^{1/(\ell-2)}$. As soon as $\ell \geq 5$, they are of length $\ll N^{1/2}$ and thus orthogonal to $\mathbf{x}, \mathbf{y}$.
- Compute a reduced basis $(\mathbf{x}', \mathbf{y}')$ of the lattice of vectors orthogonal to $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-2})$. The vectors $\mathbf{x}, \mathbf{y}$ are in this lattice, and can be recovered by a quick exhaustive search!

Lattice attack overview

- Compute a reduced basis $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-1})$ of the lattice of vectors in $\mathbb{Z}^\ell$ orthogonal to $\mathbf{v}$.
- Since $\mathbf{v} = \alpha\mathbf{x} + \beta\mathbf{y}$, the $\mathbf{b_i}$'s satisfy:

$$\alpha\langle \mathbf{b}_i, \mathbf{x}\rangle + \beta\langle \mathbf{b}_i, \mathbf{y}\rangle = 0$$

- But the smallest nonzero solution (s, t) to $\alpha s + \beta t = 0$ is of size $\approx N$, so a given $\mathbf{b}_i$ is either orthogonal to both $\mathbf{x}$ and $\mathbf{y}$, or it is of norm $> N^{1/2}$.
- Only $\ell - 2$ independent vectors orthogonal to both $\mathbf{x}$ and $\mathbf{y}$, so $\mathbf{b}_{\ell-1}$ must be of length N. Thus the remaining vectors $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-2})$ form a lattice of volume $\approx N^{3/2}/N^{1/2} = N$. Each of them is heuristically of length $\approx N^{1/(\ell-2)}$. As soon as $\ell \geq 5$, they are of length $\ll N^{1/2}$ and thus orthogonal to $\mathbf{x}, \mathbf{y}$.
- Compute a reduced basis $(\mathbf{x}', \mathbf{y}')$ of the lattice of vectors orthogonal to $(\mathbf{b}_1, \ldots, \mathbf{b}_{\ell-2})$. The vectors $\mathbf{x}, \mathbf{y}$ are in this lattice, and can be recovered by a quick exhaustive search!

Simulation results

- We can simulate this attack by picking random p, q-parts (x_i, y_i), computing the corresponding CRT values v_i in $\mathbb{Z}$ and trying to factor the modulus using just the v_i's.
- For the exhaustive search, we look for all linear combinations $s\mathbf{x}' + t\mathbf{y}'$ of $\mathbf{x}', \mathbf{y}'$ of length $< N^{1/2}$ and for each such combination, we try to factor by computing the GCD:

$$\gcd(\mathbf{v} - s\mathbf{x}' - t\mathbf{y}', N)$$

  If the linear combination is either $\mathbf{x}$ or $\mathbf{y}$, we're succesful, since $\mathbf{v}$ is congruent to $\mathbf{x}$ mod p but not mod q.
- Since $\mathbf{x}', \mathbf{y}'$ are of size $\approx N^{1/2}$, the exhaustive search has a few dozen steps at most. The full attack runs in total time < 0.01 second on a standard PC for a 1024-bit modulus.
- As predicted by the theoretical analysis, success rate is 100% for $\ell \geq 5$, regardless of modulus size. Even for $\ell = 4$ we get success rates of $\approx 40\%$.

Simulation results

- We can simulate this attack by picking random p, q-parts (x_i, y_i), computing the corresponding CRT values v_i in $\mathbb{Z}$ and trying to factor the modulus using just the v_i's.
- For the exhaustive search, we look for all linear combinations $s\mathbf{x}' + t\mathbf{y}'$ of $\mathbf{x}', \mathbf{y}'$ of length $< N^{1/2}$ and for each such combination, we try to factor by computing the GCD:

$$\gcd(\mathbf{v} - s\mathbf{x}' - t\mathbf{y}', N)$$

  If the linear combination is either $\mathbf{x}$ or $\mathbf{y}$, we're succesful, since $\mathbf{v}$ is congruent to $\mathbf{x}$ mod p but not mod q.
- Since $\mathbf{x}', \mathbf{y}'$ are of size $\approx N^{1/2}$, the exhaustive search has a few dozen steps at most. The full attack runs in total time < 0.01 second on a standard PC for a 1024-bit modulus.
- As predicted by the theoretical analysis, success rate is 100% for $\ell \geq 5$, regardless of modulus size. Even for $\ell = 4$ we get success rates of $\approx 40\%$.

Simulation results

- We can simulate this attack by picking random p, q-parts (x_i, y_i), computing the corresponding CRT values v_i in $\mathbb{Z}$ and trying to factor the modulus using just the v_i's.
- For the exhaustive search, we look for all linear combinations $s\mathbf{x}' + t\mathbf{y}'$ of $\mathbf{x}', \mathbf{y}'$ of length $< N^{1/2}$ and for each such combination, we try to factor by computing the GCD:

$$\gcd(\mathbf{v} - s\mathbf{x}' - t\mathbf{y}', N)$$

  If the linear combination is either $\mathbf{x}$ or $\mathbf{y}$, we're succesful, since $\mathbf{v}$ is congruent to $\mathbf{x} \bmod p$ but not $\bmod q$.
- Since $\mathbf{x}', \mathbf{y}'$ are of size $\approx N^{1/2}$, the exhaustive search has a few dozen steps at most. The full attack runs in total time < 0.01 second on a standard PC for a 1024-bit modulus.
- As predicted by the theoretical analysis, success rate is 100% for $\ell \geq 5$, regardless of modulus size. Even for $\ell = 4$ we get success rates of $\approx 40\%$.

Simulation results

- We can simulate this attack by picking random p, q-parts (x_i, y_i), computing the corresponding CRT values v_i in $\mathbb{Z}$ and trying to factor the modulus using just the v_i's.
- For the exhaustive search, we look for all linear combinations $s\mathbf{x}' + t\mathbf{y}'$ of $\mathbf{x}', \mathbf{y}'$ of length $< N^{1/2}$ and for each such combination, we try to factor by computing the GCD:

$$\gcd(\mathbf{v} - s\mathbf{x}' - t\mathbf{y}', N)$$

  If the linear combination is either $\mathbf{x}$ or $\mathbf{y}$, we're succesful, since $\mathbf{v}$ is congruent to $\mathbf{x}$ mod p but not mod q.
- Since $\mathbf{x}', \mathbf{y}'$ are of size $\approx N^{1/2}$, the exhaustive search has a few dozen steps at most. The full attack runs in total time < 0.01 second on a standard PC for a 1024-bit modulus.
- As predicted by the theoretical analysis, success rate is 100% for $\ell \geq 5$, regardless of modulus size. Even for $\ell = 4$ we get success rates of $\approx 40\%$.

The attack in practice

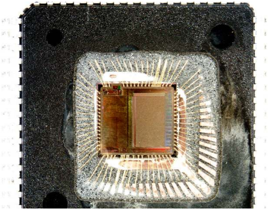
We implemented the attack against an implementation of RSA-CRT signatures on an 8-bit microcontroller.

1. Decapsulate the chip.
2. Target the SRAM and find the location of the modulus N.
3. Strike with
4. After obtaining 5 pairs of correct and faulty signatures, factor N in a fraction of a second as expected.

The attack in practice

We implemented the attack against an implementation of RSA-CRT signatures on an 8-bit microcontroller.

1. Decapsulate the chip.

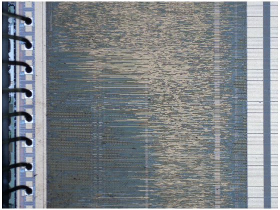


2. Target the SRAM and find the location of the modulus N.
3. Strike with
4. After obtaining 5 pairs of correct and faulty signatures, factor N in a fraction of a second as expected.

The attack in practice

We implemented the attack against an implementation of RSA-CRT signatures on an 8-bit microcontroller.

1. Decapsulate the chip.
2. Target the SRAM and find the location of the modulus N.



3. Strike with
4. After obtaining 5 pairs of correct and faulty signatures, factor N in a fraction of a second as expected.

The attack in practice

We implemented the attack against an implementation of RSA-CRT signatures on an 8-bit microcontroller.

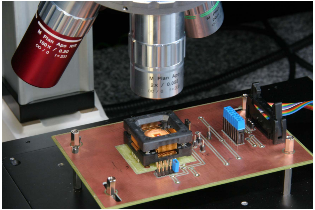
1. Decapsulate the chip.
2. Target the SRAM and find the location of the modulus N.
3. Strike with lasers!

4. After obtaining 5 pairs of correct and faulty signatures, factor N in a fraction of a second as expected.

The attack in practice

We implemented the attack against an implementation of RSA-CRT signatures on an 8-bit microcontroller.

1. Decapsulate the chip.
2. Target the SRAM and find the location of the modulus N.
3. Strike with a focused laser beam.



4. After obtaining 5 pairs of correct and faulty signatures, factor N in a fraction of a second as expected.

The attack in practice

We implemented the attack against an implementation of RSA-CRT signatures on an 8-bit microcontroller.

1. Decapsulate the chip.
2. Target the SRAM and find the location of the modulus N.
3. Strike with
4. After obtaining 5 pairs of correct and faulty signatures, factor N in a fraction of a second as expected.

Advantages and limitations

This new attack presents a number of nice features:

- Very fast.
- Only requires 5 correct/fauly signature pairs, regardless of modulus size.
- Not thwarted by standard RSA-CRT fault countermeasures such as Shamir's.

It does have some limitations:

- Needs to recover the faulty modulus N': this is a bit unrealistic in practice. However, with a few more faults of a reasonable shape, it is easy to overcome this limitation.
- Must be able to obtain a correct and a faulty signature with the same CRT value: not possible with randomized encodings.
- Most seriously: a faster, frequently used technique for CRT interpolation (Garner's formula) avoids reducing mod N altogether, and hence defeats this attack.

Advantages and limitations

This new attack presents a number of nice features:

- Very fast.
- Only requires 5 correct/fauly signature pairs, regardless of modulus size.
- Not thwarted by standard RSA-CRT fault countermeasures such as Shamir's.

It does have some limitations:

- Needs to recover the faulty modulus N': this is a bit unrealistic in practice. However, with a few more faults of a reasonable shape, it is easy to overcome this limitation.
- Must be able to obtain a correct and a faulty signature with the same CRT value: not possible with randomized encodings.
- Most seriously: a faster, frequently used technique for CRT interpolation (Garner's formula) avoids reducing mod N altogether, and hence defeats this attack.

Advantages and limitations

This new attack presents a number of nice features:

- Very fast.
- Only requires 5 correct/fauly signature pairs, regardless of modulus size.
- Not thwarted by standard RSA-CRT fault countermeasures such as Shamir's.

It does have some limitations:

- Needs to recover the faulty modulus N': this is a bit unrealistic in practice. However, with a few more faults of a reasonable shape, it is easy to overcome this limitation.
- Must be able to obtain a correct and a faulty signature with the same CRT value: not possible with randomized encodings.
- Most seriously: a faster, frequently used technique for CRT interpolation (Garner's formula) avoids reducing mod N altogether, and hence defeats this attack.

Advantages and limitations

This new attack presents a number of nice features:

- Very fast.
- Only requires 5 correct/fauly signature pairs, regardless of modulus size.
- Not thwarted by standard RSA-CRT fault countermeasures such as Shamir's.

It does have some limitations:

- Needs to recover the faulty modulus N': this is a bit unrealistic in practice. However, with a few more faults of a reasonable shape, it is easy to overcome this limitation.
- Must be able to obtain a correct and a faulty signature with the same CRT value: not possible with randomized encodings.
- Most seriously: a faster, frequently used technique for CRT interpolation (Garner's formula) avoids reducing mod N altogether, and hence defeats this attack.

Advantages and limitations

This new attack presents a number of nice features:

- Very fast.
- Only requires 5 correct/fauly signature pairs, regardless of modulus size.
- Not thwarted by standard RSA-CRT fault countermeasures such as Shamir's.

It does have some limitations:

- Needs to recover the faulty modulus N': this is a bit unrealistic in practice. However, with a few more faults of a reasonable shape, it is easy to overcome this limitation.
- Must be able to obtain a correct and a faulty signature with the same CRT value: not possible with randomized encodings.
- Most seriously: a faster, frequently used technique for CRT interpolation (Garner's formula) avoids reducing mod N altogether, and hence defeats this attack.

Advantages and limitations

This new attack presents a number of nice features:

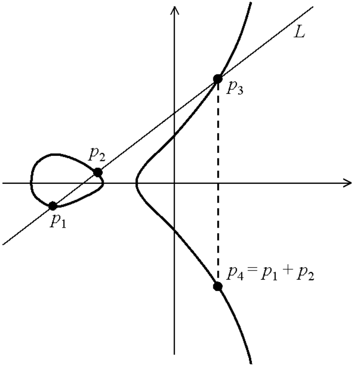
- Very fast.
- Only requires 5 correct/fauly signature pairs, regardless of modulus size.
- Not thwarted by standard RSA-CRT fault countermeasures such as Shamir's.

It does have some limitations:

- Needs to recover the faulty modulus N': this is a bit unrealistic in practice. However, with a few more faults of a reasonable shape, it is easy to overcome this limitation.
- Must be able to obtain a correct and a faulty signature with the same CRT value: not possible with randomized encodings.
- Most seriously: a faster, frequently used technique for CRT interpolation (Garner's formula) avoids reducing mod N altogether, and hence defeats this attack.

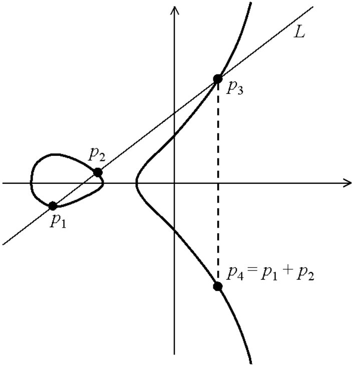
Outline

Elliptic curves



A smooth curve in the plane defined by an equation of degree 3.

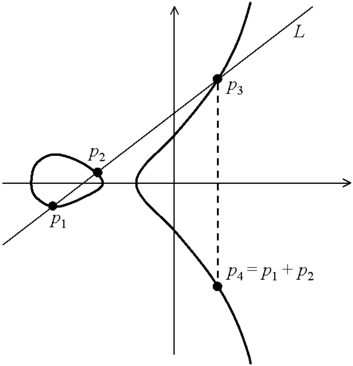
Elliptic curves



Can be put in Weierstrass form:

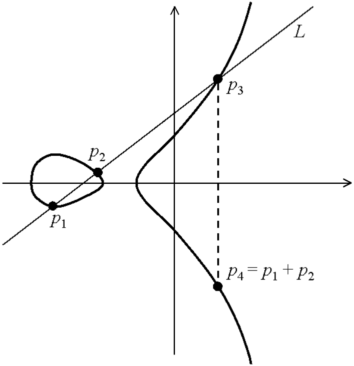
$$y^2 = x^3 + ax + b$$

Elliptic curves



Observation dating back at least to Newton: the line through two points cuts the curve at a third; if a, b are rational, the third point obtained from two rational points is also rational.
Makes it possible to define an addition law on rational points!

Elliptic curves



A central object in number theory (many important arithmetic problems from Diophantus to Wiles are about elliptic curves).

Elliptic curve cryptography

- Elliptic curves can be defined over any field, including finite fields $\mathbb{F}_q$ (we restrict attention to characteristic > 3).
- The set of $\mathbb{F}_q$-points of an elliptic curve E over $\mathbb{F}_q$ is again an abelian group $\mathbb{G} = E(\mathbb{F}_q)$ where the Discrete Logarithm Problem and Diffie-Hellman-type problems are believed to be hard ▶ suitable for cryptography! Idea due to Miller and Koblitz in the 1980s.
- In fact, the best known attack in most cases is the generic one: this means short keys and efficient protocols.
- Also come with rich structures such as pairings that don't exist in groups like $\mathbb{Z}_p^*$.

Elliptic curve cryptography

- Elliptic curves can be defined over any field, including finite fields $\mathbb{F}_q$ (we restrict attention to characteristic > 3).
- The set of $\mathbb{F}_q$-points of an elliptic curve E over $\mathbb{F}_q$ is again an abelian group $\mathbb{G} = E(\mathbb{F}_q)$ where the Discrete Logarithm Problem and Diffie-Hellman-type problems are believed to be hard ▶ suitable for cryptography! Idea due to Miller and Koblitz in the 1980s.
- In fact, the best known attack in most cases is the generic one: this means short keys and efficient protocols.
- Also come with rich structures such as pairings that don't exist in groups like $\mathbb{Z}_p^*$.

Elliptic curve cryptography

- Elliptic curves can be defined over any field, including finite fields $\mathbb{F}_q$ (we restrict attention to characteristic > 3).
- The set of $\mathbb{F}_q$-points of an elliptic curve E over $\mathbb{F}_q$ is again an abelian group $\mathbb{G} = E(\mathbb{F}_q)$ where the Discrete Logarithm Problem and Diffie-Hellman-type problems are believed to be hard ▶ suitable for cryptography! Idea due to Miller and Koblitz in the 1980s.
- In fact, the best known attack in most cases is the generic one: this means short keys and efficient protocols.
- Also come with rich structures such as pairings that don't exist in groups like $\mathbb{Z}_p^*$.

Elliptic curve cryptography

- Elliptic curves can be defined over any field, including finite fields $\mathbb{F}_q$ (we restrict attention to characteristic > 3).
- The set of $\mathbb{F}_q$-points of an elliptic curve E over $\mathbb{F}_q$ is again an abelian group $\mathbb{G} = E(\mathbb{F}_q)$ where the Discrete Logarithm Problem and Diffie-Hellman-type problems are believed to be hard ▶ suitable for cryptography! Idea due to Miller and Koblitz in the 1980s.
- In fact, the best known attack in most cases is the generic one: this means short keys and efficient protocols.
- Also come with rich structures such as pairings that don't exist in groups like $\mathbb{Z}_p^*$.

Key size comparison

Security level (bits)	RSA or $\mathbb{Z}_p^*$	Elliptic curves
80	1248	160
96	1776	192
112	2432	224
128	3248	256
256	15424	512

An ECC example: BLS signatures

- Signature scheme proposed in 2001 by Boneh, Lynn and Shacham. Achieves the shortest signature size until now.
- Public parameters: a cyclic group $\mathbb{G}$ of prime order p endowed with a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and a hash function $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$.
- KeyGen(): pick $x \overset{\$}{\leftarrow} \mathbb{Z}_p$ as the private key, and $\mathbf{P} \leftarrow [x] \cdot \mathbf{G}$ as the public key.
- Sign(m, x): compute the signature as $\mathbf{S} \leftarrow [x] \cdot \mathfrak{H}(m)$.
- Verify($m, \mathbf{S}, \mathbf{P}$): accept iff $e(\mathfrak{H}(m), \mathbf{P}) = e(\mathbf{S}, \mathbf{G})$.
- Secure if the Computational Diffie-Hellman problem is hard in $\mathbb{G}$ and $\mathfrak{H}$ is modeled as a random oracle.

An ECC example: BLS signatures

- Signature scheme proposed in 2001 by Boneh, Lynn and Shacham. Achieves the shortest signature size until now.
- Public parameters: a cyclic group $\mathbb{G}$ of prime order p endowed with a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and a hash function $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$.
- KeyGen(): pick $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ as the private key, and $\mathbf{P} \leftarrow [x] \cdot \mathbf{G}$ as the public key.
- Sign(m, x): compute the signature as $\mathbf{S} \leftarrow [x] \cdot \mathfrak{H}(m)$.
- Verify$(m, \mathbf{S}, \mathbf{P})$: accept iff $e(\mathfrak{H}(m), \mathbf{P}) = e(\mathbf{S}, \mathbf{G})$.
- Secure if the Computational Diffie-Hellman problem is hard in $\mathbb{G}$ and $\mathfrak{H}$ is modeled as a random oracle.

An ECC example: BLS signatures

- Signature scheme proposed in 2001 by Boneh, Lynn and Shacham. Achieves the shortest signature size until now.
- Public parameters: a cyclic group $\mathbb{G}$ of prime order p endowed with a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and a hash function $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$.
- KeyGen(): pick $x \xleftarrow{\$} \mathbb{Z}_p$ as the private key, and $\mathbf{P} \leftarrow [x] \cdot \mathbf{G}$ as the public key.
- Sign(m, x): compute the signature as $\mathbf{S} \leftarrow [x] \cdot \mathfrak{H}(m)$.
- Verify($m, \mathbf{S}, \mathbf{P}$): accept iff $e(\mathfrak{H}(m), \mathbf{P}) = e(\mathbf{S}, \mathbf{G})$.
- Secure if the Computational Diffie-Hellman problem is hard in $\mathbb{G}$ and $\mathfrak{H}$ is modeled as a random oracle.

An ECC example: BLS signatures

- Signature scheme proposed in 2001 by Boneh, Lynn and Shacham. Achieves the shortest signature size until now.
- Public parameters: a cyclic group $\mathbb{G}$ of prime order p endowed with a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and a hash function $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$.
- KeyGen(): pick $x \xleftarrow{\$} \mathbb{Z}_p$ as the private key, and $\mathbf{P} \leftarrow [x] \cdot \mathbf{G}$ as the public key.
- Sign(m, x): compute the signature as $\mathbf{S} \leftarrow [x] \cdot \mathfrak{H}(m)$.
- Verify$(m, \mathbf{S}, \mathbf{P})$: accept iff $e(\mathfrak{H}(m), \mathbf{P}) = e(\mathbf{S}, \mathbf{G})$.
- Secure if the Computational Diffie-Hellman problem is hard in $\mathbb{G}$ and $\mathfrak{H}$ is modeled as a random oracle.

An ECC example: BLS signatures

- Signature scheme proposed in 2001 by Boneh, Lynn and Shacham. Achieves the shortest signature size until now.
- Public parameters: a cyclic group $\mathbb{G}$ of prime order p endowed with a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and a hash function $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$.
- KeyGen(): pick $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ as the private key, and $\mathbf{P} \leftarrow [x] \cdot \mathbf{G}$ as the public key.
- Sign(m, x): compute the signature as $\mathbf{S} \leftarrow [x] \cdot \mathfrak{H}(m)$.
- Verify$(m, \mathbf{S}, \mathbf{P})$: accept iff $e(\mathfrak{H}(m), \mathbf{P}) = e(\mathbf{S}, \mathbf{G})$.
- Secure if the Computational Diffie-Hellman problem is hard in $\mathbb{G}$ and $\mathfrak{H}$ is modeled as a random oracle.

An ECC example: BLS signatures

- Signature scheme proposed in 2001 by Boneh, Lynn and Shacham. Achieves the shortest signature size until now.
- Public parameters: a cyclic group $\mathbb{G}$ of prime order p endowed with a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and a hash function $\mathfrak{H} : \{0,1\}^* \to \mathbb{G}$.
- KeyGen(): pick $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ as the private key, and $\mathbf{P} \leftarrow [x] \cdot \mathbf{G}$ as the public key.
- Sign(m, x): compute the signature as $\mathbf{S} \leftarrow [x] \cdot \mathfrak{H}(m)$.
- Verify$(m, \mathbf{S}, \mathbf{P})$: accept iff $e(\mathfrak{H}(m), \mathbf{P}) = e(\mathbf{S}, \mathbf{G})$.
- Secure if the Computational Diffie-Hellman problem is hard in $\mathbb{G}$ and $\mathfrak{H}$ is modeled as a random oracle.

Outline

Hashing to elliptic curves is a problem

- Like BLS signatures, many cryptographic protocols (for encryption, signature, PAKE, IBE, etc.) involve representing a certain numeric value, often a hash value, as an element of the group $\mathbb{G}$ where the computations occur.
- For $\mathbb{G} = \mathbb{Z}_p^*$, simply take the numeric value itself mod p.
- However, doesn't generalize when $\mathbb{G}$ is an elliptic curve group; e.g. one cannot put the value in the x-coordinate of a curve point, because only about 1/2 of possible x-values correspond to actual points.
- Elliptic curve-specific protocols have been developed to circumvent this problem (ECDSA for signature, Menezes-Vanstone for encryption, ECMQV for key agreement, etc.), but doing so with all imaginable protocols is unrealistic.

Hashing to elliptic curves is a problem

- Like BLS signatures, many cryptographic protocols (for encryption, signature, PAKE, IBE, etc.) involve representing a certain numeric value, often a hash value, as an element of the group $\mathbb{G}$ where the computations occur.
- For $\mathbb{G} = \mathbb{Z}_p^*$, simply take the numeric value itself $\bmod p$.
- However, doesn't generalize when $\mathbb{G}$ is an elliptic curve group; e.g. one cannot put the value in the x-coordinate of a curve point, because only about 1/2 of possible x-values correspond to actual points.
- Elliptic curve-specific protocols have been developed to circumvent this problem (ECDSA for signature, Menezes-Vanstone for encryption, ECMQV for key agreement, etc.), but doing so with all imaginable protocols is unrealistic.

Hashing to elliptic curves is a problem

- Like BLS signatures, many cryptographic protocols (for encryption, signature, PAKE, IBE, etc.) involve representing a certain numeric value, often a hash value, as an element of the group $\mathbb{G}$ where the computations occur.
- For $\mathbb{G} = \mathbb{Z}_p^*$, simply take the numeric value itself $\bmod p$.
- However, doesn't generalize when $\mathbb{G}$ is an elliptic curve group; e.g. one cannot put the value in the x-coordinate of a curve point, because only about 1/2 of possible x-values correspond to actual points.
- Elliptic curve-specific protocols have been developed to circumvent this problem (ECDSA for signature, Menezes-Vanstone for encryption, ECMQV for key agreement, etc.), but doing so with all imaginable protocols is unrealistic.

Hashing to elliptic curves is a problem

- Like BLS signatures, many cryptographic protocols (for encryption, signature, PAKE, IBE, etc.) involve representing a certain numeric value, often a hash value, as an element of the group $\mathbb{G}$ where the computations occur.
- For $\mathbb{G} = \mathbb{Z}_p^*$, simply take the numeric value itself $\bmod p$.
- However, doesn't generalize when $\mathbb{G}$ is an elliptic curve group; e.g. one cannot put the value in the x-coordinate of a curve point, because only about 1/2 of possible x-values correspond to actual points.
- Elliptic curve-specific protocols have been developed to circumvent this problem (ECDSA for signature, Menezes-Vanstone for encryption, ECMQV for key agreement, etc.), but doing so with all imaginable protocols is unrealistic.

A naive approach

- We have reasonable construction of hash functions to bit strings, or to a group like $\mathbb{Z}_p$.
- Hence, a naive approach to hashing to an elliptic curve group $\mathbb{G}$ of order p could be to start from a hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{Z}_p$ and simply define:

$$\mathfrak{H}(m) = [\mathfrak{h}(m)] \cdot \mathbf{G}$$

- This is a bad idea. Taking BLS signatures as an example, the signature on a message m can the be written as:

$$\mathbf{S} = [x] \cdot \mathfrak{H}(m) = \left[x\mathfrak{h}(m)\right] \cdot \mathbf{G} = [\mathfrak{h}(m)] \cdot \mathbf{P}$$

  and hence computed publicly. Completely breaks security!
- So we have to be careful.

A naive approach

- We have reasonable construction of hash functions to bit strings, or to a group like $\mathbb{Z}_p$.
- Hence, a naive approach to hashing to an elliptic curve group $\mathbb{G}$ of order p could be to start from a hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{Z}_p$ and simply define:

$$\mathfrak{H}(m) = [\mathfrak{h}(m)] \cdot \mathbf{G}$$

- This is a bad idea. Taking BLS signatures as an example, the signature on a message m can the be written as:

$$\mathbf{S} = [x] \cdot \mathfrak{H}(m) = \left[x\mathfrak{h}(m)\right] \cdot \mathbf{G} = [\mathfrak{h}(m)] \cdot \mathbf{P}$$

and hence computed publicly. Completely breaks security!

- So we have to be careful.

A naive approach

- We have reasonable construction of hash functions to bit strings, or to a group like $\mathbb{Z}_p$.
- Hence, a naive approach to hashing to an elliptic curve group $\mathbb{G}$ of order p could be to start from a hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{Z}_p$ and simply define:

$$\mathfrak{H}(m) = [\mathfrak{h}(m)] \cdot \mathbf{G}$$

- This is a bad idea. Taking BLS signatures as an example, the signature on a message m can the be written as:

$$\mathbf{S} = [x] \cdot \mathfrak{H}(m) = \left[x\mathfrak{h}(m)\right] \cdot \mathbf{G} = [\mathfrak{h}(m)] \cdot \mathbf{P}$$

  and hence computed publicly. Completely breaks security!
- So we have to be careful.

A naive approach

- We have reasonable construction of hash functions to bit strings, or to a group like $\mathbb{Z}_p$.
- Hence, a naive approach to hashing to an elliptic curve group $\mathbb{G}$ of order p could be to start from a hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{Z}_p$ and simply define:

$$\mathfrak{H}(m) = [\mathfrak{h}(m)] \cdot \mathbf{G}$$

- This is a bad idea. Taking BLS signatures as an example, the signature on a message m can the be written as:

$$\mathbf{S} = [x] \cdot \mathfrak{H}(m) = \left[x\mathfrak{h}(m)\right] \cdot \mathbf{G} = [\mathfrak{h}(m)] \cdot \mathbf{P}$$

  and hence computed publicly. Completely breaks security!
- So we have to be careful.

The traditional solution

- Start from a hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q$ to the base field.
- For k bits of security:
  1. concatenate the message m with a counter c from 0 to $k-1$;
  2. initialize the counter as 0;
  3. if the hash value $x = \mathfrak{h}(c \| m)$ is a valid x-coordinate on the curve (i.e. $x^3 + ax + b$ is a square in $\mathbb{F}_q$), return one of the two corresponding points as $\mathfrak{H}(m)$; otherwise increment the counter and try again.
- The probability of a concatenated value being valid is $1/2 + O(1/\sqrt{q})$, so k iterations ensure k bits of security.
- Problem: this does not run in constant time. Can facilitate side-channel attacks, especially for protocols like PAKE.

The traditional solution

- Start from a hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q$ to the base field.
- For k bits of security:
    1. concatenate the message m with a counter c from 0 to $k-1$;
    2. initialize the counter as 0;
    3. if the hash value $x = \mathfrak{h}(c \| m)$ is a valid x-coordinate on the curve (i.e. $x^3 + ax + b$ is a square in $\mathbb{F}_q$), return one of the two corresponding points as $\mathfrak{H}(m)$; otherwise increment the counter and try again.
- The probability of a concatenated value being valid is $1/2 + O(1/\sqrt{q})$, so k iterations ensure k bits of security.
- Problem: this does not run in constant time. Can facilitate side-channel attacks, especially for protocols like PAKE.

The traditional solution

- Start from a hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q$ to the base field.
- For k bits of security:
  1. concatenate the message m with a counter c from 0 to $k-1$;
  2. initialize the counter as 0;
  3. if the hash value $x = \mathfrak{h}(c \| m)$ is a valid x-coordinate on the curve (i.e. $x^3 + ax + b$ is a square in $\mathbb{F}_q$), return one of the two corresponding points as $\mathfrak{H}(m)$; otherwise increment the counter and try again.
- The probability of a concatenated value being valid is $1/2 + O(1/\sqrt{q})$, so k iterations ensure k bits of security.
- Problem: this does not run in constant time. Can facilitate side-channel attacks, especially for protocols like PAKE.

The traditional solution

- Start from a hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q$ to the base field.
- For k bits of security:
  1. concatenate the message m with a counter c from 0 to $k-1$;
  2. initialize the counter as 0;
  3. if the hash value $x = \mathfrak{h}(c \| m)$ is a valid x-coordinate on the curve (i.e. $x^3 + ax + b$ is a square in $\mathbb{F}_q$), return one of the two corresponding points as $\mathfrak{H}(m)$; otherwise increment the counter and try again.
- The probability of a concatenated value being valid is $1/2 + O(1/\sqrt{q})$, so k iterations ensure k bits of security.
- Problem: this does not run in constant time. Can facilitate side-channel attacks, especially for protocols like PAKE.

The traditional solution

- Start from a hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q$ to the base field.
- For k bits of security:
  1. concatenate the message m with a counter c from 0 to $k-1$;
  2. initialize the counter as 0;
  3. if the hash value $x = \mathfrak{h}(c \| m)$ is a valid x-coordinate on the curve (i.e. $x^3 + ax + b$ is a square in $\mathbb{F}_q$), return one of the two corresponding points as $\mathfrak{H}(m)$; otherwise increment the counter and try again.
- The probability of a concatenated value being valid is $1/2 + O(1/\sqrt{q})$, so k iterations ensure k bits of security.
- Problem: this does not run in constant time. Can facilitate side-channel attacks, especially for protocols like PAKE.

The traditional solution

- Start from a hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q$ to the base field.
- For k bits of security:
  1. concatenate the message m with a counter c from 0 to $k-1$;
  2. initialize the counter as 0;
  3. if the hash value $x = \mathfrak{h}(c \| m)$ is a valid x-coordinate on the curve (i.e. $x^3 + ax + b$ is a square in $\mathbb{F}_q$), return one of the two corresponding points as $\mathfrak{H}(m)$; otherwise increment the counter and try again.
- The probability of a concatenated value being valid is $1/2 + O(1/\sqrt{q})$, so k iterations ensure k bits of security.
- Problem: this does not run in constant time. Can facilitate side-channel attacks, especially for protocols like PAKE.

The traditional solution

- Start from a hash function $\mathfrak{h} : \{0,1\}^* \to \mathbb{F}_q$ to the base field.
- For k bits of security:
  1. concatenate the message m with a counter c from 0 to $k-1$;
  2. initialize the counter as 0;
  3. if the hash value $x = \mathfrak{h}(c \| m)$ is a valid x-coordinate on the curve (i.e. $x^3 + ax + b$ is a square in $\mathbb{F}_q$), return one of the two corresponding points as $\mathfrak{H}(m)$; otherwise increment the counter and try again.
- The probability of a concatenated value being valid is $1/2 + O(1/\sqrt{q})$, so k iterations ensure k bits of security.
- Problem: this does not run in constant time. Can facilitate side-channel attacks, especially for protocols like PAKE.

The Boneh-Franklin construction

For their elliptic curve-based IBE scheme [BF01], Boneh and Franklin introduced the following hash function construction.

They use *supersingular* elliptic curves, of the form:

$$y^2 = x^3 + b$$

over $\mathbb{F}_q$ with $q \equiv 2 \pmod 3$. Admit the following deterministic encoding:

$$f : u \mapsto \left((u^2 - b)^{1/3}, u\right)$$

Solves the problem: efficient, constant-time, quasi-bijective and secure ▶ if $\mathfrak{h}$ is a good hash function to $\mathbb{F}_q$, $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ is well-behaved: has the properties of a RO to the curve if $\mathfrak{h}$ is modeled as a RO to $\mathbb{F}_q$.The IBE scheme is secure for $\mathfrak{H}$ in the ROM for $\mathfrak{h}$.

Downside: limited to supersingular curves.

The Boneh-Franklin construction

For their elliptic curve-based IBE scheme [BF01], Boneh and Franklin introduced the following hash function construction.

They use *supersingular* elliptic curves, of the form:

$$y^2 = x^3 + b$$

over $\mathbb{F}_q$ with $q \equiv 2 \pmod 3$. Admit the following deterministic encoding:

$$f : u \mapsto \left((u^2 - b)^{1/3}, u\right)$$

Solves the problem: efficient, constant-time, quasi-bijective and secure ▶ if $\mathfrak{h}$ is a good hash function to $\mathbb{F}_q$, $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ is well-behaved: has the properties of a RO to the curve if $\mathfrak{h}$ is modeled as a RO to $\mathbb{F}_q$.The IBE scheme is secure for $\mathfrak{H}$ in the ROM for $\mathfrak{h}$.

Downside: limited to supersingular curves.

The Boneh-Franklin construction

For their elliptic curve-based IBE scheme [BF01], Boneh and Franklin introduced the following hash function construction.

They use *supersingular* elliptic curves, of the form:

$$y^2 = x^3 + b$$

over $\mathbb{F}_q$ with $q \equiv 2 \pmod 3$. Admit the following deterministic encoding:

$$f : u \mapsto \left((u^2 - b)^{1/3}, u\right)$$

Solves the problem: efficient, constant-time, quasi-bijective and secure ▶ if $\mathfrak{h}$ is a good hash function to $\mathbb{F}_q$, $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ is well-behaved: has the properties of a RO to the curve if $\mathfrak{h}$ is modeled as a RO to $\mathbb{F}_q$. The IBE scheme is secure for $\mathfrak{H}$ in the ROM for $\mathfrak{h}$.

Downside: limited to supersingular curves.

The Boneh-Franklin construction

For their elliptic curve-based IBE scheme [BF01], Boneh and Franklin introduced the following hash function construction.

They use *supersingular* elliptic curves, of the form:

$$y^2 = x^3 + b$$

over $\mathbb{F}_q$ with $q \equiv 2 \pmod 3$. Admit the following deterministic encoding:

$$f : u \mapsto \left((u^2 - b)^{1/3}, u\right)$$

Solves the problem: efficient, constant-time, quasi-bijective and secure ▶ if $\mathfrak{h}$ is a good hash function to $\mathbb{F}_q$, $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ is well-behaved: has the properties of a RO to the curve if $\mathfrak{h}$ is modeled as a RO to $\mathbb{F}_q$.The IBE scheme is secure for $\mathfrak{H}$ in the ROM for $\mathfrak{h}$.

Downside: limited to supersingular curves.

Ordinary curves: Icart

At CRYPTO 2009, Icart presented a construction for ordinary curves when $q \equiv 2 \pmod 3$. Generalization of the supersingular case.

Defined as $f\colon u \mapsto (x, y)$ with

$$x = \left(v^2 - b - \frac{u^6}{27}\right)^{1/3} + \frac{u^2}{3} \qquad y = ux + v \qquad v = \frac{3a - u^4}{6u}$$

Efficient, constant-time, and applies to almost all elliptic curves. *However*, image size is only $\approx 5/8$ of all points. The construction $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ is easily distinguished from a RO to the curve even if $\mathfrak{h}$ is modeled as a RO. ▶ Security?

Many more deterministic encodings to ordinary curves proposed recently, but with the same limitation.

Ordinary curves: Icart

At CRYPTO 2009, Icart presented a construction for ordinary curves when $q \equiv 2 \pmod 3$. Generalization of the supersingular case.

Defined as $f\colon u \mapsto (x, y)$ with

$$x = \left(v^2 - b - \frac{u^6}{27}\right)^{1/3} + \frac{u^2}{3} \qquad y = ux + v \qquad v = \frac{3a - u^4}{6u}$$

Efficient, constant-time, and applies to almost all elliptic curves. *However*, image size is only $\approx 5/8$ of all points. The construction $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ is easily distinguished from a RO to the curve even if $\mathfrak{h}$ is modeled as a RO. ▶ Security?

Many more deterministic encodings to ordinary curves proposed recently, but with the same limitation.

Ordinary curves: Icart

At CRYPTO 2009, Icart presented a construction for ordinary curves when $q \equiv 2 \pmod 3$. Generalization of the supersingular case.

Defined as $f\colon u \mapsto (x, y)$ with

$$x = \left(v^2 - b - \frac{u^6}{27}\right)^{1/3} + \frac{u^2}{3} \qquad y = ux + v \qquad v = \frac{3a - u^4}{6u}$$

Efficient, constant-time, and applies to almost all elliptic curves. *However*, image size is only $\approx 5/8$ of all points. The construction $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ is easily distinguished from a RO to the curve even if $\mathfrak{h}$ is modeled as a RO. ▶ Security?

Many more deterministic encodings to ordinary curves proposed recently, but with the same limitation.

Outline

Security in the ROM

Is it secure to use $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ as a hash function to the curve?

More precisely: if a scheme is proved secure assuming $\mathfrak{H}$ is a RO, is the security preserved if one instantiates $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ with $\mathfrak{h}$ modeled as a RO?

- For a number of schemes: yes (related to random self-reducibility properties of the underlying security assumptions).
- In general: no, security breaks down (ad-hoc counter-examples).
- Difficult to give a simple criterion for the security proof to go through.
- Can we propose constructions that will work all the time instead?

Security in the ROM

Is it secure to use $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ as a hash function to the curve?

More precisely: if a scheme is proved secure assuming $\mathfrak{H}$ is a RO, is the security preserved if one instantiates $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ with $\mathfrak{h}$ modeled as a RO?

- For a number of schemes: yes (related to random self-reducibility properties of the underlying security assumptions).
- In general: no, security breaks down (ad-hoc counter-examples).
- Difficult to give a simple criterion for the security proof to go through.
- Can we propose constructions that will work all the time instead?

Security in the ROM

Is it secure to use $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ as a hash function to the curve?

More precisely: if a scheme is proved secure assuming $\mathfrak{H}$ is a RO, is the security preserved if one instantiates $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ with $\mathfrak{h}$ modeled as a RO?

- For a number of schemes: yes (related to random self-reducibility properties of the underlying security assumptions).
- In general: no, security breaks down (ad-hoc counter-examples).
- Difficult to give a simple criterion for the security proof to go through.
- Can we propose constructions that will work all the time instead?

Security in the ROM

Is it secure to use $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ as a hash function to the curve?

More precisely: if a scheme is proved secure assuming $\mathfrak{H}$ is a RO, is the security preserved if one instantiates $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ with $\mathfrak{h}$ modeled as a RO?

- For a number of schemes: yes (related to random self-reducibility properties of the underlying security assumptions).
- In general: no, security breaks down (ad-hoc counter-examples).
- Difficult to give a simple criterion for the security proof to go through.
- Can we propose constructions that will work all the time instead?

Security in the ROM

Is it secure to use $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ as a hash function to the curve?

More precisely: if a scheme is proved secure assuming $\mathfrak{H}$ is a RO, is the security preserved if one instantiates $\mathfrak{H}(m) = f(\mathfrak{h}(m))$ with $\mathfrak{h}$ modeled as a RO?

- For a number of schemes: yes (related to random self-reducibility properties of the underlying security assumptions).
- In general: no, security breaks down (ad-hoc counter-examples).
- Difficult to give a simple criterion for the security proof to go through.
- Can we propose constructions that will work all the time instead?

Indifferentiability

High-level formulation of our problem: find a condition under which an ideal primitive (the RO to the curve) can be replaced by a construction based on another ideal primitive (a RO to $\mathbb{F}_q$) so that all security proof are preserved.

Answer: indifferentiability (Maurer et al., 2004). Roughly speaking, the construction is indifferentiable from the primitive if no PPT adversary can tell them apart with non-negligible probability.

But this is a bit abstract. Easy to test criterion for a hash function construction to work?

Indifferentiability

High-level formulation of our problem: find a condition under which an ideal primitive (the RO to the curve) can be replaced by a construction based on another ideal primitive (a RO to $\mathbb{F}_q$) so that all security proof are preserved.

Answer: indifferentiability (Maurer et al., 2004). Roughly speaking, the construction is indifferentiable from the primitive if no PPT adversary can tell them apart with non-negligible probability.

But this is a bit abstract. Easy to test criterion for a hash function construction to work?

Admissible encodings

We consider hash function constructions of the form:

$$\mathfrak{H}(m) = F(\mathfrak{h}(m))$$

where $\mathfrak{h}$ is modeled as a RO to a some set S (easy to hash to) and F is a deterministic function $S \to E(\mathbb{F}_q)$.

We can prove that $\mathfrak{H}$ is indifferentiable from a RO to $E(\mathbb{F}_q)$ as soon as the function F is admissible in the following sense:

- **Computable** in deterministic polynomial time;
- **Regular** for s uniformly distributed in S, the distribution of $F(s)$ is statistically indistinguishable from the uniform distribution in $E(\mathbb{F}_q)$;
- **Samplable** there is a PPT algorithm which for any $\varpi \in E(\mathbb{F}_q)$ returns an uniformly distributed element in $F^{-1}(\varpi)$.

Admissible encodings

We consider hash function constructions of the form:

$$\mathfrak{H}(m) = F(\mathfrak{h}(m))$$

where $\mathfrak{h}$ is modeled as a RO to a some set S (easy to hash to) and F is a deterministic function $S \to E(\mathbb{F}_q)$.

We can prove that $\mathfrak{H}$ is indifferentiable from a RO to $E(\mathbb{F}_q)$ as soon as the function F is admissible in the following sense:

Computable in deterministic polynomial time;

Regular for s uniformly distributed in S, the distribution of $F(s)$ is statistically indistinguishable from the uniform distribution in $E(\mathbb{F}_q)$;

Samplable there is a PPT algorithm which for any $\varpi \in E(\mathbb{F}_q)$ returns an uniformly distributed element in $F^{-1}(\varpi)$.

Remarks

- We can quantify precisely the “loss” in random oracle security when instantiating $\mathfrak{H}$ in this manner (in terms of the statistical distance between $F(s)$ and uniform, and the running time of the sampling algorithm).
- Icart’s function is *not* admissible: computable and samplable, but not regular.
- A construction like $\mathfrak{H}(m) = [\mathfrak{h}(m)] \cdot \mathbf{G}$ is *not* admissible: computable and regular but not samplable.

Remarks

- We can quantify precisely the "loss" in random oracle security when instantiating $\mathfrak{H}$ in this manner (in terms of the statistical distance between $F(s)$ and uniform, and the running time of the sampling algorithm).
- Icart's function is *not* admissible: computable and samplable, but not regular.
- A construction like $\mathfrak{H}(m) = [\mathfrak{h}(m)] \cdot \mathbf{G}$ is *not* admissible: computable and regular but not samplable.

Remarks

- We can quantify precisely the "loss" in random oracle security when instantiating $\mathfrak{H}$ in this manner (in terms of the statistical distance between $F(s)$ and uniform, and the running time of the sampling algorithm).
- Icart's function is *not* admissible: computable and samplable, but not regular.
- A construction like $\mathfrak{H}(m) = [\mathfrak{h}(m)] \cdot \mathbf{G}$ is *not* admissible: computable and regular but not samplable.

General construction

E ordinary elliptic curve over $\mathbb{F}_q$, $\mathbf{G}$ generator of $E(\mathbb{F}_q)$ (assumed cyclic of cardinality N) and $f\colon \mathbb{F}_q \to E(\mathbb{F}_q)$ deterministic encoding like Icart's function.

Under mild assumptions on f (verified for all deterministic encodings proposed so far), the following is an admissible function $\mathbb{F}_q \times \mathbb{Z}/N\mathbb{Z} \to E(\mathbb{F}_q)$:

$$F(u, v) = f(u) + [v] \cdot \mathbf{G}$$

Thus, $\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + [\mathfrak{h}_2(m)] \cdot \mathbf{G}$ is indifferentiable from a RO, in the ROM for $\mathfrak{h}_1, \mathfrak{h}_2$.

Downside: quite inefficient (≈ 10 times slower than Icart's function alone).

Proof sketch

The function F is:

Computable Clearly.

Regular With v uniformly distributed in $\mathbb{Z}/N\mathbb{Z}$ it is clear that $f(u) + [v] \cdot \mathbf{G}$ is uniformly distributed in $E(\mathbb{F}_q)$, regardless of the behavior of f.

Samplable To sample $F^{-1}(\mathbf{P})$, pick a random $v \in \mathbb{Z}/N\mathbb{Z}$ and solve the algebraic equation $f(u) = \mathbf{P} - [v] \cdot \mathbf{G}$ for u. For Icart, there are at most 4 solutions, easy to enumerate. Return (u, v) for one of those solutions u at random, or try again if there are none.

Efficient construction

A much more efficient construction of an admissible encoding is as follows:

$$F(u, v) = f(u) + f(v)$$

where f is Icart's function.

Thus, $\mathfrak{H}(m) = f(\mathfrak{h}_1(m)) + f(\mathfrak{h}_2(m))$ is indifferentiable from a RO, in the ROM for $\mathfrak{h}_1, \mathfrak{h}_2$.

Only requires two evaluations of Icart's function, so quite efficient. No restriction on the curve.

Downside: proof is more difficult.

More precisely, computability and samplability are proved like before. The hard part is regularity: showing that the cardinality of $F^{-1}(\mathbf{P})$ is almost constant along the curve.

Proof idea

We want to show that the number of solutions $(u, v) \in (\mathbb{F}_q)^2$ to the equation $f(u) + f(v) = \mathbf{P}$ is constant up to negligible deviations when $\mathbf{P}$ varies along the curve (possibly with a few exceptions).

Key idea: the set of solutions (u, v) forms a curve in the plane. The Hasse-Weil bound ensures that such a curve always has $q + O(\sqrt{q})$ points. QED.

Technical difficulties:

- Icart's function f is not a morphism, only an algebraic correspondence. The correct geometric pictures involves a curve C with morphisms $h: C \to E$ and $p: C \to \mathbb{P}^1$ such that $f = h \circ p^{-1}$.
- Show that $s: C \times C \to E$ is geometrically "nice", except at a few exceptional points (to be found and dealt with).
- Show that the preimage of "nice" points is indeed an irreducible curve on $C \times C$. Compute its genus (it's 49).

Proof idea

We want to show that the number of solutions $(u, v) \in (\mathbb{F}_q)^2$ to the equation $f(u) + f(v) = \mathbf{P}$ is constant up to negligible deviations when $\mathbf{P}$ varies along the curve (possibly with a few exceptions).

Key idea: the set of solutions (u, v) forms a curve in the plane. The Hasse-Weil bound ensures that such a curve always has $q + O(\sqrt{q})$ points. QED.

Technical difficulties:

- Icart's function f is not a morphism, only an algebraic correspondence. The correct geometric pictures involves a curve C with morphisms $h\colon C \to E$ and $p\colon C \to \mathbb{P}^1$ such that $f = h \circ p^{-1}$.
- Show that $s\colon C \times C \to E$ is geometrically "nice", except at a few exceptional points (to be found and dealt with).
- Show that the preimage of "nice" points is indeed an irreducible curve on $C \times C$. Compute its genus (it's 49).

Summary and outlook

- Consider the instantiations of random oracles in elliptic curve-based cryptosystems;
- Suggest a framework for constructing well-behaved hash functions to ordinary elliptic curves;
- Propose two such constructions, one more general, the other more efficient.

Further problems:

- Extend the efficient construction to any constant-time encoding to elliptic and hyperelliptic curves (done!)
- Construct *injective encodings* to ordinary curves (some progress)
- Understand how the possibility of encoding scalars as curve points affects elliptic curve-based protocols (wide open)

Summary and outlook

- Consider the instantiations of random oracles in elliptic curve-based cryptosystems;
- Suggest a framework for constructing well-behaved hash functions to ordinary elliptic curves;
- Propose two such constructions, one more general, the other more efficient.

Further problems:

- Extend the efficient construction to any constant-time encoding to elliptic and hyperelliptic curves (done!)
- Construct *injective encodings* to ordinary curves (some progress)
- Understand how the possibility of encoding scalars as curve points affects elliptic curve-based protocols (wide open)

Summary and outlook

- Consider the instantiations of random oracles in elliptic curve-based cryptosystems;
- Suggest a framework for constructing well-behaved hash functions to ordinary elliptic curves;
- Propose two such constructions, one more general, the other more efficient.

Further problems:

- Extend the efficient construction to any constant-time encoding to elliptic and hyperelliptic curves (done!)
- Construct *injective encodings* to ordinary curves (some progress)
- Understand how the possibility of encoding scalars as curve points affects elliptic curve-based protocols (wide open)

Summary and outlook

- Consider the instantiations of random oracles in elliptic curve-based cryptosystems;
- Suggest a framework for constructing well-behaved hash functions to ordinary elliptic curves;
- Propose two such constructions, one more general, the other more efficient.

Further problems:

- Extend the efficient construction to any constant-time encoding to elliptic and hyperelliptic curves (done!)
- Construct *injective encodings* to ordinary curves (some progress)
- Understand how the possibility of encoding scalars as curve points affects elliptic curve-based protocols (wide open)

Summary and outlook

- Consider the instantiations of random oracles in elliptic curve-based cryptosystems;
- Suggest a framework for constructing well-behaved hash functions to ordinary elliptic curves;
- Propose two such constructions, one more general, the other more efficient.

Further problems:

- Extend the efficient construction to any constant-time encoding to elliptic and hyperelliptic curves (done!)
- Construct *injective encodings* to ordinary curves (some progress)
- Understand how the possibility of encoding scalars as curve points affects elliptic curve-based protocols (wide open)

Summary and outlook

- Consider the instantiations of random oracles in elliptic curve-based cryptosystems;
- Suggest a framework for constructing well-behaved hash functions to ordinary elliptic curves;
- Propose two such constructions, one more general, the other more efficient.

Further problems:

- Extend the efficient construction to any constant-time encoding to elliptic and hyperelliptic curves (done!)
- Construct *injective encodings* to ordinary curves (some progress)
- Understand how the possibility of encoding scalars as curve points affects elliptic curve-based protocols (wide open)

Contributions to RSA cryptanalysis (I)

- **Fault attacks**
  - Fault Attacks Against EMV Signatures
    Coron, Naccache, T. [CT-RSA 2010]
  - Modulus Fault Attacks Against RSA Signatures
    Brier, Naccache, Nguyen, T. [CHES 2011; JCEN]
  - Lattice-Based Fault Attacks on Signatures
    Nguyen, T. [FAC]

Contributions to RSA cryptanalysis (II)

- **Attacks of ad-hoc paddings**
  - Practical Cryptanalysis of ISO/IEC 9796-2 and EMV Signatures
    Coron, Naccache, T., Weinmann [CRYPTO 2009]
  - On the Broadcast and Validity-Checking Security of PKCS#1 v1.5
    Bauer, Coron, Naccache, T., Vergnaud [ACNS 2010]
  - Another Look at RSA Signatures With Affine Padding
    Coron, Naccache, T. [submitted]
- **Other contributions**
  - Factoring Unbalanced Moduli with Known Bits
    Brier, Naccache, T. [ICISC 2009]
  - Cryptanalysis of the RSA Subgroup Assumption from TCC 2005
    Coron, Joux, Naccache, Mandal, T. [PKC 2011]

Contributions to ECC (I)

- **Hashing and encoding**

Estimating the Size of the Image of Deterministic Hash Functions to Elliptic Curves
Fouque, T. [LATINCRYPT 2010]

Efficient Indifferentiable Hashing into Ordinary Elliptic Curves
Brier, Coron, Icart, Madore, Randriam, T. [CRYPTO 2010]

Deterministic Encoding and Hashing to Odd Hyperelliptic Curves
Fouque, T. [Pairing 2010]

Securing E-passports with Elliptic Curves
Chabanne, T. [IEEE Security & Privacy]

Indifferentiable Deterministic Hashing to Elliptic and Hyperelliptic Curves
Farashahi, Fouque, Shparlinski, T., Voloch
[to appear in Math. Comp.]

Contributions to ECC (II)

- **Other contributions**
  - Huff's Model for Elliptic Curves
    Joye, T., Vergnaud [ANTS-IX]
  - A Nagell Algorithm in Any Characteristic
    T. [Festschrift JJQ]

Other areas

- **Fully-homomorphic encryption**
  - Fully Homomorphic Encryption over the Integers with Shorter Public Keys
    Coron, Mandal, Naccache, T. [CRYPTO 2011]
  - Optimization of Fully Homomorphic Encryption
    Coron, Naccache, T. [submitted]
- **Prime generation**
  - Close to Uniform Prime Number Generation With Fewer Random Bits
    Fouque, T. [submitted]

Thank you!